A METHOD OF MANAGING A GRAPHICAL USER INTERFACE

The invention relates to a method of managing keyboard events of a graphical user interface configured in the form of a tree of graphical elements.

The invention also relates to a system including this kind of interface.

The field of the invention is that of navigating the graphical interface of a system which has a keyboard but no pointing tool or mouse, such as a mobile telephone or a pocket organizer.

BACKGROUND OF THE INVENTION

Graphical applications executed on this type of system can be developed in various programming languages and in particular in an object-oriented programming language such as Java.

In object-oriented programming, classes are defined which each have their own characteristics. The classes are related by parent-child relationships, the child class inheriting characteristics from the class of its parent.

Most graphical applications developed in the Java environment define classes representing graphical elements or components that can be nested. Examples of graphical components are: windows, input fields, "OK" buttons, labels, etc. The programmer can also define new components.

Figure 1a) shows one example of a screen E composed in this way. The screen E includes simple graphical components such as input fields, labels, and buttons. It also includes a component (container) C that is complex, i.e. one in which components are nested. A window-type graphical component "Window 1" has been chosen first. Three components are nested in the "Window 1" component: an input field "Input 1", an "OK" button and a container C in which are nested an input field "Input 2" and a label "Label 1".

Figure 1b) shows a tree of graphical components

which corresponds to the above composition. The parent graphical component "WINDOW 1" has three child graphical components "INPUT 1", "OK" and a container C, the last of which itself has two child components "INPUT 2" and "LABEL 1".

It is useful for the user of this kind of system to be provided with keyboard shortcuts. A keyboard shortcut consists of a key which, when pressed, or a combination of keys which, when pressed simultaneously, initiate a specific action regardless of the graphical component on which it is located. For example, the keyboard shortcut "Ctrl Z" entails pressing the "Ctrl" key and the "Z" key simultaneously. Thus in this example the user may require a keyboard shortcut to initiate the action "Go back to the previous screen" when the component that is active, i.e. awaiting a keyboard event, is the "INPUT 2" component. The user would also wish to have access to the same keyboard shortcut, initiating the same action, if the active component were the "OK" button.

However, the navigation processes of graphical applications developed in Java do not provide for the use of keyboard shortcuts.

OBJECT AND SUMMARY OF THE INVENTION

The object of the invention is to circumvent this limitation for applications executing on systems with no mouse by enabling developers of such applications to define keyboard shortcuts.

The invention provides a method of managing keyboard events for navigating a graphical user interface configured in the form of a tree of graphical elements EG_i, wherein each graphical element EG_i of the tree is associated with a list LEG_i of keys TG_j and wherein each key TG_j listed in said lists is associated with an action A_j to be initiated on receipt of a keyboard event corresponding to said key and said graphical element.

If a graphical element EG_i is active and a keyboard event is detected by the interface, the method includes

the following steps:

- comparing said keyboard event to the keys TG_j listed in said lists LEG_i, starting with the list for the active component EG_i and working back up said tree, and
- initiating the action associated with the first key corresponding to said keyboard event.

The invention also relates to a portable system DP having a graphical interface including a keyboard CL, a screen E and an interface management unit GI, wherein the management unit employs a method of managing keyboard events as previously described.

The system DP can be a mobile telephone or a pocket organizer.

BRIEF DESCRIPTION OF THE DRAWINGS

Other features and advantages of the invention will become clearly apparent on reading the description given by way of non-limiting example and with reference to the accompanying drawings, in which:

- Figure 1a) shows one example of a screen made up of graphical elements in accordance with the prior art,
- Figure 1b) shows a tree structure associated with the above example,
- Figure 2a) shows a tree of graphical elements in accordance with the invention,
- Figure 2b) shows in more detail a graphical element which is associated in accordance with the invention with a list of keyboard shortcut pairs and the corresponding actions,
- Figure 3 is a flowchart showing how the invention works, and
- Figure 4 shows a portable system employing a keyboard event management method according to the invention.

MORE DETAILED DESCRIPTION

Figure 2a) shows a tree of graphical elements in accordance with the invention in which each graphical element EG_i, where i varies in the range from 1 to the

number N of graphical elements used to compose a screen, is associated with a list LEG_i of pairs, shown in Figure 2b), each of which pairs comprises a keyboard shortcut TG_j and an associated action A_i to be executed, where j varies in the range from 1 to the number M of pairs defined in the tree.

In the mode of operation shown in Figure 3, i.e. if a keyboard event occurs when the component EG_i is active, the keyboard event is compared to each keyboard shortcut TG_j in the list LEG_i for the active component EG_i. If the event is not in that list, the comparison is repeated but this time using the list for the component preceding the active component in the tree, i.e. the parent component. The process therefore works back up the tree in this way until it finds the keyboard shortcut TG_j corresponding to the keyboard event. The action associated with the keyboard shortcut TG_j is then initiated.

Because of this path through the tree, it is not essential to define a pair (TG_j, A_j) in the list for a component for that pair to be associated with that component: it is sufficient for it to be defined in the list of one of the "parent" components. Thus a list LEG_i can be empty. If, after following the path through the tree, no keyboard shortcut corresponding to the keyboard event is found, then no action is initiated; an error can be indicated to the user.

This global definition of the pairs (TG_j, A_j) has a number of advantages. If new components EG_i, or even new containers, are introduced into the tree subsequently, they automatically inherit the pairs (TG_j, A_j) from their parents, without it being necessary to associate them with the latter in their list LEG_i. This shortens the lists commensurately.

This mode of operation based on a path through the tree offers great flexibility. It may be required in connection with a component EG_i to change a pair in the list LEG_l for example the action A_k associated with an

existing global shortcut TG_k. It is sufficient to define in the list LEG_1 a new pair associating a new action A_k, with the keyboard shortcut TG_k.

As shown in Figure 4, a system DP employing this kind of graphical interface conventionally includes a keyboard CL, a screen E and a user interface control unit GI included in control electronics EC. The developer responsible for management of the interface integrates the proposed solution into the management unit GI.